Type-based Proxy Re-encryption and its Construction

Qiang Tang

Faculty of EWI, University of Twente, the Netherlands
q.tang@utwente.nl

Abstract. Recently, the concept of proxy re-encryption has been shown very useful in a number of applications, especially in enforcing access control policies. In existing proxy re-encryption schemes, the delegatee can decrypt all ciphertexts for the delegator after re-encryption by the proxy. Consequently, in order to implement fine-grained access control policies, the delegator needs to either use multiple key pairs or trust the proxy to behave honestly. In this paper, we extend this concept and propose type-based proxy re-encryption, which enables the delegator to selectively delegate his decryption right to the delegatee while only needs one key pair. As a result, type-based proxy re-encryption enables the delegator to implement fine-grained policies with one key pair without any additional trust on the proxy. We provide a security model for our concept and provide formal definitions for semantic security and ciphertext privacy which is a valuable attribute in privacy-sensitive contexts. We propose two type-based proxy re-encryption schemes: one is CPA secure with ciphertext privacy while the other is CCA secure without ciphertext privacy.

1 Introduction

In a proxy re-encryption scheme [5,15], a delegator (say, Alice) and a delegatee (say, Bob) generate a proxy key that allows a semi-trusted third party (say, the proxy) to convert ciphertexts encrypted under Alice's public key into ciphertexts which can be decrypted by Bob. Recently, proxy re-encryption has been shown very useful in a number of applications such as access control in file storage [1], email forwarding [19], and law enforcement [13].

Motivation. We have the following observations on the existing proxy re-encryption schemes and their applications. One is that, with respect to one key pair of the delegator, the proxy is able to re-encrypt all the ciphertexts so that the delegatee can obtain all the plaintexts. The other is that, in many applications, it is likely that the delegator only wishes a specific delegatee to see a subset of his messages. In order to implement fine-grained access control policies, the delegator can choose a different key pair for each possible subset of his messages and choose a proxy to delegate his decryption right. However, this approach is infeasible in practice. Alternatively, the delegator can choose to trust the proxy to

enforce his policies by re-encrypting the pre-defined subset of his ciphertexts to the specific delegatee. This approach is also infeasible because of the strong trust requirement on the proxy. For example, if the proxy colludes with a malicious delegatee (or, the proxy is compromised), then all messages of the delegator will be compromised.

In fact, the delegator might have more concerns about the delegation in practical applications. An observation is that, with a public key encryption scheme, if Alice encrypts a message for Bob using his public key, then Alice can stay anonymous. However, with a proxy re-encryption scheme, this kind of anonymity might not be trivially achieved. For example, in the schemes in [11], Bob can (at least) tell whether messages are from the same user or not.

Contribution. In this paper, we propose the concept of type-based proxy re-encryption which enables the delegator to selectively delegate his decryption right to delegatees. In a type-based proxy re-encryption scheme, the delegator categorizes his messages (ciphertexts) into different subsets and is capable of delegating the decryption right of each subset to a specific delegatee. The ciphertexts for the delegator are generated based on the delegator's public key and the message type which is used to identify the message subset. This new primitive has (at least) the following promising properties. One is that the delegator only needs one key pair so that the key management problem is simplified. The other is that the delegator can choose a particular proxy for a specific delegatee, which might be based on the sensitiveness of the delegation. Compromise of one proxy key will only affect one subset of messages.

We provide a security model for our concept and provide formal definitions for semantic security and ciphertext privacy which is a valuable attribute in privacy-sensitive contexts. If a type-based proxy re-encryption scheme achieves ciphertext privacy property then all re-encrypted ciphertexts are indistinguishable from normal ciphertexts originally generated for the delegatee, therefore, message senders (the delegators) remain anonymous. We propose two type-based proxy re-encryption schemes. The first scheme achieves ciphertext privacy and is IND-PR-CPA secure based on the XDH and the Co-BDH assumptions in the random oracle model. The second scheme is IND-PR-CCA secure based on the BDH and the KE assumptions in the random oracle model, but it does not achieve ciphertext privacy.

Related work. Mambo and Okamoto [15] firstly propose the concept of delegation of decryption right in the context of speeding up decryption operations. Blaze, Bleumer and Strauss [5] introduce the concept of atomic proxy cryptography which is proxy re-encryption. They present an Elgamal [10] based proxy re-encryption scheme, in which the proxy is also capable of converting ciphertexts encrypted for Bob into ciphertexts which can be decrypted by Alice. Jakobsson [14] and Zhou *et al.* [20] simultaneously propose quorum-based protocols, which divide the proxy into many components. Dodis and Ivan [13] propose generic constructions of proxy re-encryption schemes by using double-encryption. Ateniese *et al.* [1] propose an Elgamal based scheme and show its application in

securing file systems. In addition, Ateniese *et al.* also point out a number of desirable properties for proxy re-encryption schemes. Note that these papers are mainly focused on the traditional public-key encryption schemes. Apart from the generic construction of Dodis and Ivan [13], there are two identity-based proxy re-encryption schemes: one is proposed by Green and Ateniese [11] and the other is proposed by Matsuo [16]. In both schemes, the delegator and the delegatee are assumed to be registered at the same domain (or, the same key generation center).

Organization. The rest of the paper is organized as follows. In Section 2 we provide some preliminary knowledge. In Section 3 we present the security model for type-based proxy re-encryption. In Section 4 we present the IND-PR-CPA secure type-based proxy re-encryption scheme with ciphertext privacy and prove its security. In Section 5 we present the IND-PR-CCA secure type-based proxy re-encryption scheme without ciphertext privacy and prove its security. In Section 6 we conclude the paper.

2 Preliminaries

If x is chosen uniformly at random from the set Y, then we write $x \in_R Y$. The symbol $\perp$ denotes an error message. Our security analysis is done in the random oracle model [4]. Next, we review the necessary knowledge about pairing and the related assumptions, and then review the Public Key Encryption (PKE).

2.1 Review of Pairing

More detailed information can be found in [6,12]. Generally, a pairing (or, bilinear map) satisfies the following properties:

1. $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are three multiplicative groups of prime order p;
2. g_1 is a generator of $\mathbb{G}_1$ and g_2 is a generator of $\mathbb{G}_2$;
3. $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is an efficiently-computable bilinear map with the following properties:
   - Bilinear: for all $a, b \in \mathbb{Z}_p$, we have $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$.
   - Non-degenerate: $\hat{e}(g_1, g_2) \neq 1$.

The Co-Bilinear Diffie-Hellman (Co-BDH) problem is as follows: given $g_1, g_1^a, g_1^b \in \mathbb{G}_1, g_2, g_2^c \in \mathbb{G}_2$ as input, output $\hat{e}(g_1, g_2)^{abc} \in \mathbb{G}_T$. An algorithm $\mathcal{A}$ has advantage ϵ in solving Co-BDH in $\mathbb{G}$ if

$$\Pr[\mathcal{A}(g_1, g_2, g_1^a, g_1^b, g_2^c) = \hat{e}(g_1, g_2)^{abc}] \geq \epsilon,$$

where the probability is over the random choice of $a, b, c \in \mathbb{Z}_p$, and the random bits of $\mathcal{A}$.

Definition 1. *We say that the Co-BDH assumption holds if any polynomial-time adversary has only a negligible advantage ϵ in solving the Co-BDH problem.*

Note that the Co-BDH assumption is closely related to the Asymmetric Bilinear Diffie-Hellman (ABDH) assumption defined in [7].

The eXternal Diffie-Hellman (XDH) assumption is also widely used in the literature (e.g. [8]). We say that an algorithm $\mathcal{A}$ has advantage ϵ in solving the XDH problem if

$$|\Pr[\mathcal{A}(g_2, g_2^a, g_2^b, g_2^{a \cdot b}) = 0] - \Pr[\mathcal{A}(g_2, g_2^a, g_2^b, g_2^c) = 0]| \geq \epsilon,$$

where the probability is over the random choice of $a, b, c \in \mathbb{Z}_p$, and the random bits of $\mathcal{A}$.

Definition 2. *We say that the XDH assumption holds if any polynomial-time adversary has only a negligible advantage ϵ in solving the XDH problem.*

Instead of the above general setting, the simplified setting is also widely used (e.g. [6]). Here, a pairing (or, bilinear map) satisfies the following properties:

1. $\mathbb{G}$ and $\mathbb{G}_T$ are two multiplicative groups of prime order p;
2. g is a generator of $\mathbb{G}$;
3. $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is an efficiently-computable bilinear map with the following properties:
   - Bilinear: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, we have $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$.
   - Non-degenerate: $\hat{e}(g, g) \neq 1$.

The Bilinear Diffie-Hellman (BDH) problem is as follows: given a tuple $g, g^a, g^b, g^c \in \mathbb{G}$ as input, output $\hat{e}(g, g)^{abc} \in \mathbb{G}$. An algorithm $\mathcal{A}$ has advantage ϵ in solving BDH if

$$\Pr[\mathcal{A}(g, g^a, g^b, g^c) = \hat{e}(g, g)^{abc}] \geq \epsilon,$$

where the probability is over the random choice of $a, b, c \in \mathbb{Z}_p$, and the random bits of $\mathcal{A}$.

Definition 3. *We say that the BDH assumption holds if any polynomial-time algorithm has only a negligible advantage ϵ in solving the BDH problem.*

Besides these computational/decisional assumptions, the Knowledge of Exponent (KE) assumption is also used in a number of papers (e.g. [3,9]). The KE assumption is defined as follows.

Definition 4. *For any adversary $\mathcal{A}$, which takes a KE challenge (g, g^a) as input and returns (C, Y) where $Y = C^a$, there exists an extractor $\mathcal{A}'$, which takes the same input as $\mathcal{A}$ returns c such that $g^c = C$.*

2.2 Review of Public Key Encryption

A Public Key Encryption (PKE) scheme [17] involves a Trusted Third Party (TTP) and users, and consists of four algorithms ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}$) which are defined as follows. As a standard practice, the security of a PKE scheme is evaluated by an attack game played between a challenger and an adversary, where the challenger simulates the protocol execution and answers the oracle queries from the adversary. Corresponding to the PKE algorithms, we also introduce the oracles for the adversary.

- $\mathsf{Setup}(k)$: Run by the TTP, this algorithm takes a security parameter k as input and generates the public parameter $params$, which is an implicit input for other algorithms and we omit it in the description for simplicity.
- $\mathsf{KeyGen}(k)$: Run by a user, this algorithm generates a key pair (pk, sk). An KeyGen oracle can be queried with a public key pk; the challenger returns the corresponding private key sk.
- $\mathsf{Encrypt}(m, pk)$: Run by the message sender, this algorithm takes a message m and a public key pk as input, and outputs a ciphertext c encrypted under the public key pk.
- $\mathsf{Decrypt}(c, sk)$: Run by the message receiver, this algorithm takes a ciphertext c and the private key sk as input, and outputs the message m. A $\mathsf{Decrypt}$ oracle can be queried with a pair (c, pk) as input; the challenger returns $\mathsf{Decrypt}(c, sk)$.

We extend the concept of PKE to type-based PKE which enables a message sender to explicitly include some type information in the encryption process. A type-based PKE consists of four algorithms ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}$), where Setup and KeyGen are defined as above, and

- $\mathsf{Encrypt}(m, t, pk)$: Run by the message sender, this algorithm takes a message m, a type string t, and a public key pk as input, and outputs a ciphertext c encrypted under the public key pk. Note that both c and t should be sent to the message receiver.
- $\mathsf{Decrypt}(c, t, sk)$: Run by the message receiver, this algorithm takes a ciphertext c, a message type t, and the private key sk as input, and outputs the message m. A $\mathsf{Decrypt}$ oracle can be queried with a pair (c, t, pk) as input; the challenger returns $\mathsf{Decrypt}(c, t, sk)$.

In the above descriptions, the type information t can also be included as a part of the ciphertext c, but we have explicitly used type information t as a label of the ciphertext c. This is only a description preference.

3 The concept of Type-based Proxy Re-encryption

In a type-based proxy re-encryption scheme, the delegator possesses a key pair (pk, sk) with a type-based PKE scheme $(\mathsf{Setup}_1, \mathsf{KeyGen}_1, \mathsf{Encrypt}_1, \mathsf{Decrypt}_1)$,

as defined in Section 2.2. We assume that the delegatees use a PKE scheme $(\mathsf{Setup}_2, \mathsf{KeyGen}_2, \mathsf{Encrypt}_2, \mathsf{Decrypt}_2)$.

Suppose the delegator wants to delegate his decryption right for messages with type t to a delegatee with key pair (pk', sk'), he runs the $\mathsf{Pextract}$ algorithm to generate the proxy key.

- $\mathsf{Pextract}(pk, pk', t, sk)$: This algorithm takes the delegator's public key pk, the delegatee's public key pk', a message type t, the delegator's private key sk as input and outputs the delegation key $rk_{pk \xrightarrow{t} pk'}$. A $\mathsf{Pextract}$ oracle can be queried with a tuple (pk, pk', t) as input; the challenger returns the proxy key $rk_{pk \xrightarrow{t} pk'}$.

Note that all proxy keys, $rk_{pk \xrightarrow{t} pk'}$ for any t and pk', are generated based on the delegator's single key pair. To delegate his right, the delegator assigns $rk_{pk \xrightarrow{t} pk'}$ to an appropriate proxy, which will preform the re-encryption for the delegator's ciphertexts.

- $\mathsf{Preenc}(c, t, rk_{pk \xrightarrow{t} pk'})$: Run by the proxy, this algorithm takes a ciphertext c (for the delegator), a message type t, and the proxy key $rk_{pk \xrightarrow{t} pk'}$ as input, and outputs a new ciphertext c' (for the delegatee with (pk', sk')). A Preenc oracle can be queried with (c, t, pk, pk') as input; the challenger returns $\mathsf{Preenc}(c, t, rk_{pk \xrightarrow{t} pk'})$.

In contrast to the assumption of multi-level delegation (e.g. in [11]), we assume that there is only one level delegation, namely the delegatees will not further delegate their decryption rights to other users.

3.1 Threat Model for Type-based Proxy Re-encryption

In practice, there might be multiple different parties acting as proxies. For example, Alice may choose Proxy 1 to delegate her decryption right to Bob and choose a proxy 2 to delegate his decryption right to Charlie, where these two proxies have no relationship. Every involved proxy is assumed to be semi-honest in the following sense: it will honestly convert the delegator's ciphertexts using the proxy key; however, it might act maliciously to obtain some information about the plaintexts of the delegator and the delegatee.

We identify the following security requirements with respect to the semantic security for plaintexts.

1. Firstly, the proxy should not obtain any information about the plaintexts of either the delegator or the delegatee.
2. Secondly, the delegatee should be able to decrypt all the appropriate type of plaintexts of the delegator after the re-encryption by the proxy. However, the delegatee alone should not obtain any information about the plaintexts before the re-encrypted by the proxy. This is essential when we want the proxy to be a policy enforcer.

In our formal definitions, these requirements lead to the IND-PR-CCA/IND-PR-CPA security of type-based proxy encryption schemes for the delegator.

With a public key encryption scheme, if Alice encrypts a message to Bob using his public key, then Alice can stay anonymous. However, with a proxy re-encryption scheme, this kind of anonymity might not be trivially achieved. For example, in the schemes in [11], Bob can tell whether messages are from the same user or not. In many potential applications of proxy re-encryption, this might become a privacy concern if the delegator does not want the delegatee to know whether a certain message is from him or not. Therefore, a re-encrypted ciphertext should be indistinguishable from a normal ciphertext generated under the delegatee's public key. This requirement leads to the definition of ciphertext privacy.

3.2 Formal Security Definitions

Before the formal definitions, we first introduce the idea behind our IND-PR-CCA definition (the idea of IND-PR-CPA security follows immediately). The definition covers two types of adaptive adversaries: a malicious delegatee and a malicious proxy. In the case of a malicious delegatee with key pair (pk', sk'), the adversary is allowed to know all proxy keys for message types $t \neq t^*$, either the private key sk'' or the proxy key $rk_{pk \xrightarrow{t^*} pk''}$ for any other delegatee with (pk'', sk''), but not $rk_{pk \xrightarrow{t^*} pk'}$. In the case of a malicious proxy with $rk_{pk \xrightarrow{t^*} pk'}$, the adversary is allowed to know all proxy keys for message types $t \neq t^*$, either the private key sk'' or the proxy key $rk_{pk \xrightarrow{t^*} pk''}$ for any other delegatee with (pk'', sk''), but not sk'. In addition, the adversary is capable of issuing decryption queries (see our remarks below). We believe the adversary has been granted the most privileges possible.

IND-PR-CCA security. We first define the semantic security against a chosen ciphertext attack for the delegator.

Definition 5. *A type-based proxy re-encryption scheme is said to be IND-PR-CCA secure for the delegator if any polynomial time adversary has only a negligible advantage in the IND-PR-CCA game, where the adversary's advantage is defined to be* $|\Pr[b' = b] - \frac{1}{2}|$.

Analogous to the IND-CCA definition [2], as depicted in Figure 1, the IND-PR-CCA game is as follows.

1. Game setup: The challenger takes a security parameter k as input, runs Setup_1 to generate the public system parameter $params_1$ and runs Setup_2 to generate the public system parameter $params_2$. The challenger runs KeyGen_1 to generate a key pair (pk, sk).
2. Phase 1: The adversary takes $params_1$, $params_2$, and pk as input, and has access to the following types of oracles: KeyGen_2, $\mathsf{Pextract}$, Preenc, $\mathsf{Decrypt}_1$, and $\mathsf{Decrypt}_2$. Once the adversary decides that Phase 1 is over, it outputs two

1. $params_1 \xleftarrow{\$} \mathsf{Setup}_1(k); params_2 \xleftarrow{\$} \mathsf{Setup}_2(k); (pk, sk) \xleftarrow{\$} \mathsf{KeyGen}_1(k)$
2. $(m_0, m_1, t^*) \xleftarrow{\$} \mathcal{A}^{(\mathsf{KeyGen}_2, \mathsf{Pextract}, \mathsf{Preenc}, \mathsf{Decrypt}_1, \mathsf{Decrypt}_2)}(params_1, params_2, pk)$
3. $b \xleftarrow{\$} \{0,1\}; c_b \xleftarrow{\$} \mathsf{Encrypt}_1(m_b, t^*, pk)$
4. $b' \xleftarrow{\$} \mathcal{A}^{(\mathsf{KeyGen}_2, \mathsf{Pextract}, \mathsf{Preenc}, \mathsf{Decrypt}_1, \mathsf{Decrypt}_2)}(params_1, params_2, pk, c_b)$

Fig. 1. IND-PR-CCA Security

equal length plaintexts m_0, m_1, and a message type t^*. At the end of Phase 1, the following constraint should be satisfied: For any pk', if (pk, pk', t^*) has been queried to the $\mathsf{Pextract}$ oracle, then pk' should not have been queried to the KeyGen_2 oracle.

3. Challenge: The challenger picks a random bit $b \in \{0,1\}$ and returns $c_b = \mathsf{Encrypt}_1(m_b, t^*, pk)$ as the challenge to the adversary.
4. Phase 2: The adversary is allowed to continue querying the same types of oracles as in Phase 1. At the end of Phase 2, we have the following constraints.
   (a) (c_b, t^*, pk) has not been queried to the $\mathsf{Decrypt}_1$ oracle.
   (b) For any pk', if (pk, pk', t^*) has been queried to the $\mathsf{Pextract}$ oracle, then pk' should not have been queried to the KeyGen_2 oracle.
   (c) If pk' has been queried to the KeyGen_2 oracle, (c_b, t^*, pk, pk') should not have been queried to the Preenc oracle.
   (d) If (pk, pk', t^*) has been queried to the $\mathsf{Pextract}$ oracle, then (c'_b, pk') should not have been queried to the $\mathsf{Decrypt}_2$ oracle where c'_b is a valid output of $\mathsf{Preenc}(c_b, t^*, pk, pk')$.
5. Guess (game ending): The adversary outputs a guess $b' \in \{0,1\}$.

We remark on the constraints (c) and (d) in the above game. In the case of type-based proxy re-encryption, if the adversary obtains sk', then a Preenc query with the input (c_b, t^*, pk, pk') is equivalent to a $\mathsf{Decrypt}_1$ query with the input (c_b, t^*, pk). If the adversary obtains $rk_{pk \xrightarrow{t^*} pk'}$, then a $\mathsf{Decrypt}_2$ query with the input (c'_b, pk'), where c'_b is a valid output of a Preenc query with the input (c_b, t^*, pk, pk'), is equivalent to a $\mathsf{Decrypt}_1$ query with the input (c_b, t^*, pk). These constraints are necessary to prevent the adversary from winning the game trivially.

IND-PR-CPA security. We define the semantic security against a chosen plaintext attack for the delegator. Analogous to the IND-CPA definitions for traditional PKE schemes [2], we can just remove the decryption privileges of the adversary to define the IND-PR-CPA game, i.e. the oracle accesses to Preenc, $\mathsf{Decrypt}_1$, and $\mathsf{Decrypt}_2$ are removed. However, we need to provide the adversary a $\mathsf{Preenc}^\dagger$ oracle to cover the case that a malicious delegatee with (pk', sk') can always decrypt the re-encrypted ciphertexts for him.

– $\mathsf{Preenc}^\dagger(m, t, pk, pk')$: the challenger returns $\mathsf{Preenc}(\mathsf{Encrypt}_1(m, t, pk), t, rk_{pk \xrightarrow{t} pk'})$.

1. $params_1 \xleftarrow{\$} \mathsf{Setup}_1(k); params_2 \xleftarrow{\$} \mathsf{Setup}_2(k); (pk, sk) \xleftarrow{\$} \mathsf{KeyGen}_1(k)$
2. $(m_0, m_1, t^*) \xleftarrow{\$} \mathcal{A}^{(\mathsf{KeyGen}_2, \mathsf{Pextract}, \mathsf{Preenc}^\dagger)}(params_1, params_2, pk)$
3. $b \xleftarrow{\$} \{0,1\}; c_b \xleftarrow{\$} \mathsf{Encrypt}_1(m_b, t^*, pk)$
4. $b' \xleftarrow{\$} \mathcal{A}^{(\mathsf{KeyGen}_2, \mathsf{Pextract}, \mathsf{Preenc}^\dagger)}(params_1, params_2, pk, c_b)$

Fig. 2. IND-PR-CPA Security

Note that the re-encrypted ciphertext might leak some information about the delegator's private key and hence help the adversary to obtain some information of the delegator's ciphertexts. This has been ignored in [11].

The IND-PR-CPA game is depicted in Figure 2, and a detailed description can be obtained by forbidding the Preenc, $\mathsf{Decrypt}_1$, and $\mathsf{Decrypt}_2$ oracle access and providing $\mathsf{Preenc}^\dagger$ oracle access in the IND-PR-CCA game.

Definition 6. *A type-based proxy re-encryption scheme is said to be IND-PR-CPA secure for the delegator if any polynomial time adversary has only a negligible advantage in the IND-PR-CPA game (depicted in Figure 2), where the adversary's advantage is defined to be* $|\Pr[b' = b] - \frac{1}{2}|$.

Ciphertext privacy. The ciphertext privacy attribute means that, for any delegatee, a re-encrypted ciphertext is indistinguishable from a normal ciphertext generated under the delegatee's public key. The attack game for ciphertext privacy is as follows.

1. Game setup: The challenger takes a security parameter k as input, runs Setup_1 to generate the public system parameter $params_1$ and runs Setup_2 to generate the public system parameter $params_2$. The challenger runs KeyGen_1 to generate a key pair (pk, sk).
2. Phase 1: The adversary takes $params_1$, $params_2$, and pk as input, and has access to the following types of oracles: KeyGen_2 and $\mathsf{Pextract}$. Once the adversary decides that Phase 1 is over, it outputs a message m, a message type t, and pk'.
3. Challenge: The challenger picks a random bit $b \in \{0,1\}$ and returns a challenge c_b as follows.
   - If $b = 0$, then $c_b = \mathsf{Encrypt}_2(m, pk')$.
   - If $b = 1$, then $c_b = \mathsf{Preenc}(\mathsf{Encrypt}_1(m, t, pk), t, rk_{pk \xrightarrow{t} pk'})$.
4. Phase 2: The adversary has access to the same types of oracles as in Phase 1.
5. Guess (game ending): The adversary outputs a guess $b' \in \{0,1\}$.

Definition 7. *A type-based proxy re-encryption scheme achieves ciphertext privacy if any polynomial time adversary has only a negligible advantage in the above game, where the adversary's advantage is defined to be* $|\Pr[b' = b] - \frac{1}{2}|$.

4 CPA-secure Scheme with Ciphertext Privacy

In this section we propose a new type-based proxy re-encryption scheme which achieves ciphertext privacy. The scheme is IND-PR-CPA secure based on the Co-BDH and the XDH assumptions in the random oracle model. We note that the ciphertext privacy is achieved through the re-randomization by the proxy.

4.1 Description of the Scheme

The delegator's type-based PRE scheme. The delegator uses the following type-based PKE scheme $(\mathsf{Setup}_1, \mathsf{KeyGen}_1, \mathsf{Encrypt}_1, \mathsf{Decrypt}_1)$.

1. $\mathsf{Setup}_1(k)$: This algorithm generates three multiplicative cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ of prime order p, a random generator g_1 of $\mathbb{G}_1$, a random generator g_2 of $\mathbb{G}_2$, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, and two hash functions

$$\mathsf{H}_1 : \{0,1\}^* \to \mathbb{G}_2, \ \ \mathsf{H}_2 : \{0,1\}^* \to \{0,1\}^\ell,$$

   where ℓ is a polynomial in k and $\{0,1\}^\ell$ is the plaintext space. The public parameter is denoted as $params_1 = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, \mathsf{H}_1, \mathsf{H}_2, \hat{e}, \ell)$, and we further assume the type information is $t \in \{0,1\}^*$.
2. $\mathsf{KeyGen}_1(k)$: This algorithm outputs a key pair (pk, sk) where $u \in_R \mathbb{Z}_p$, $pk = g_1^u$, and $sk = u$.
3. $\mathsf{Encrypt}_1(m, t, pk)$: This algorithm outputs a ciphertext $c = (c_1, c_2, c_3)$, where

$$r \in_R \mathbb{Z}_p, \ \ c_1 = g_1^r, \ \ h \in_R \mathbb{G}_T, \ \ c_2 = m \oplus \mathsf{H}_2(h).$$

$$\begin{aligned} c_3 &= h \cdot \hat{e}(pk, \mathsf{H}_1(0||t))^r \\ &= h \cdot \hat{e}(g_1, \mathsf{H}_1(0||t))^{u \cdot r} \end{aligned}$$

4. $\mathsf{Decrypt}_1(c, t, sk)$: This algorithm recovers m as follows:

$$\begin{aligned} m' &= c_2 \oplus \mathsf{H}_2\left(\frac{c_3}{\hat{e}(c_1, \mathsf{H}_1(0||t))^{sk}}\right) \\ &= m \oplus \mathsf{H}_2(h) \oplus \mathsf{H}_2\left(\frac{h \cdot \hat{e}(g_1, \mathsf{H}_1(0||t))^{u \cdot r}}{\hat{e}(g_1, \mathsf{H}_1(0||t))^{u \cdot r}}\right) \\ &= m \end{aligned}$$

The delegatee's PRE scheme. The delegatees use the following PKE scheme $(\mathsf{Setup}_2, \mathsf{KeyGen}_2, \mathsf{Encrypt}_2, \mathsf{Decrypt}_2)$.

1. $\mathsf{Setup}_2(k)$: Set $params_2 = params_1 = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, \mathsf{H}_1, \mathsf{H}_2, \hat{e}, \ell)$.
2. $\mathsf{KeyGen}_2(k)$: This algorithm outputs a key pair (pk, sk), where $v \in_R \mathbb{Z}_p$, $pk = g_2^v$, and $sk = v$.

3. $\mathsf{Encrypt}_2(m, pk)$: This algorithm outputs a ciphertext $c = (c_0, c_1, c_2, c_3)$, where

$$x, y \in_R \mathbb{Z}_p,\ \ c_0 = g_1^x,\ \ c_1 = g_2^{v \cdot y},\ \ w \in_R \mathbb{G}_T,\ \ c_2 = m \oplus \mathsf{H}_2(w),$$

$$\begin{aligned} c_3 &= w \cdot \hat{e}(g_1^x, g_2^y) \\ &= w \cdot \hat{e}(g_1, g_2)^{x \cdot y} \end{aligned}$$

4. $\mathsf{Decrypt}_2(c, sk)$: This algorithm recovers m as follows:

$$\begin{aligned} m' &= c_2 \oplus \mathsf{H}_2(\frac{c_3}{\hat{e}(c_0, c_1)^{v^{-1}}}) \\ &= m \oplus \mathsf{H}_2(w) \oplus \mathsf{H}_2(\frac{w \cdot \hat{e}(g_1, g_2)^{x \cdot y}}{\hat{e}(g_1^x, g_2^{v \cdot y})^{v^{-1}}}) \\ &= m \end{aligned}$$

The delegation algorithms. Suppose the delegator has a key pair (pk, sk), where $pk = g_1^u$ and $sk = u$. Suppose a delegatee has a key pair (pk', sk'), where $pk' = g_2^v$ and $sk' = v$. The algorithms $\mathsf{Pextract}$ and Preenc are defined as follows.

– $\mathsf{Pextract}(pk, pk', t, sk)$: The proxy key is $rk_{pk \xrightarrow{t} pk'}$, where

$$s \in_R \mathbb{Z}_p,\ \ rk_{pk \xrightarrow{t} pk'} = (g_2^{v \cdot s}, g_2^s \cdot \mathsf{H}_1(0||t)^{-sk}).$$

– $\mathsf{Preenc}(c, t, rk_{pk \xrightarrow{t} pk'})$: Given a ciphertext $c = (c_1, c_2, c_3)$, where

$$c_1 = g_1^r,\ \ c_2 = m \oplus \mathsf{H}_2(h),\ \ c_3 = h \cdot \hat{e}(g_1, \mathsf{H}_1(0||t))^{u \cdot r},$$

this algorithm computes a new ciphertext $c' = (c'_0, c'_1, c'_2, c'_3)$, where

$$z \in_R \mathbb{Z}_p,\ \ c'_0 = c_1,\ \ c'_1 = g_2^{v \cdot (s+z)},\ \ c'_2 = c_2,$$

$$\begin{aligned} c'_3 &= c_3 \cdot \hat{e}(c'_0, g_2^{(s+z)} \cdot \mathsf{H}_1(0||t)^{-sk}) \\ &= h \cdot \hat{e}(g_1^r, \mathsf{H}_1(0||t)^u) \cdot \hat{e}(g_1^r, g_2^{(s+z)} \cdot \mathsf{H}_1(0||t)^{-u}) \\ &= h \cdot \hat{e}(g_1, g_2)^{r \cdot (s+z)}. \end{aligned}$$

The re-encrypted ciphertext is also a valid ciphertext for the delegatee so that the delegatee can obtain the plaintext m by running $\mathsf{Decrypt}_2$.

4.2 Security Analysis

The proposed scheme is proven to be IND-PR-CPA secure from Lemma 1 and achieve ciphertext privacy from Lemma 2.

Lemma 1. *The proposed type-based proxy re-encryption scheme is IND-PR-CPA secure based on the Co-BDH and the XDH assumptions in the random oracle model.*

Proof sketch. We suppose that the total number of queries of the form $0||t$ issued to H_1 in Phase 1 of the IND-PR-CPA game is bounded by integer q_1[1]. Suppose an adversary $\mathcal{A}$ has the advantage ϵ in the IND-PR-CPA game. The security proof is done through a sequence of games [18].

Game_0: In this game, the challenger faithfully answers the oracle queries from $\mathcal{A}$. The challenger simulates the random oracle H_1 as follows: the challenger maintains a list of vectors, each of them containing a request message, an element of $\mathbb{G}_2$ (the hash-code for this message), and an element of $\mathbb{Z}_p$. After receiving a request message, the challenger first checks its list to see whether the request message is already in the list. If the check succeeds, the challenger returns the stored element of $\mathbb{G}_2$; otherwise, the challenger returns g_2^x, where x a randomly chosen element of $\mathbb{Z}_p$, and stores the new vector in the list. The challenger simulates the random oracle H_2 as follows: the challenger maintains a list of vectors, each of them containing a request message and an element of $\{0,1\}^\ell$ (the hash-code for this message). After receiving a request message, the challenger first checks its list to see whether the request message is already in the list. If the check succeeds, the challenger returns the stored element of $\{0,1\}^\ell$; otherwise, the challenger returns y which is a randomly chosen element of $\{0,1\}^\ell$, and stores the new vector in the list. Let $\delta_0 = \Pr[b' = b]$ in Game_0, as we assumed at the beginning, $|\delta_0 - \frac{1}{2}| = \epsilon$.

Game_1: In this game, the challenger performs as follows.

1. Game setup: The challenger faithfully performs.
2. Phase 1: The challenger randomly selects $\tau \in \{1, 2, \cdots, q_1 + 1\}$. If $\tau \leq q_1$, suppose the τ-th input to H_1 is $0||t'$. The challenger faithfully answers the oracle queries.
3. Challenge: After receiving (m_0, m_1, t^*) from the adversary, the challenger aborts if one of the following events occurs:
   (a) $0||t^*$ has been queried to H_1 but is not the τ-th query.
   (b) $0||t^*$ has not been queried to H_1 but $\tau \leq q_1$.
   The challenger faithfully returns the challenge.
4. The challenger performs faithfully in Phase 2.

Note that if the challenger does not abort and the τ-th queried has been made in Phase 1, then $t' = t^*$. The probability that the challenger successfully ends is (at least) $\frac{1}{q_1+1}$, i.e. the probability that the challenger does not abort in its execution is $\frac{1}{q_1+1}$. Let $\delta_1 = \Pr[b' = b]$ given that the challenger successfully ends, in which case $\delta_1 = \delta_0$. Let θ_1 be the probability that the challenger successfully ends and $b' = b$ in Game_1, then we have $\theta_1 = \frac{\delta_1}{q_1+1}$.

[1] For simplicity of description, it is reasonable to assume that the total number is counted for queries with different inputs.

Game_2: In this game, the challenger performs as follows.

1. Game setup: The challenger faithfully performs.
2. Phase 1: The challenger performs in the same way as in Game_1 except for the following.
   - $\mathsf{Pextract}(pk, pk', t')$: The proxy key is computed as $rk_{pk \xrightarrow{t'} pk'}$, where

$$s \in_R \mathbb{Z}_p,\ \ q \in_R \mathbb{G}_2,\ \ rk_{pk \xrightarrow{t'} pk'} = (q, g_2^s \cdot \mathsf{H}_1(0||t')^{-sk}).$$

   - $\mathsf{Preenc}^{\dagger}(m, t', pk, pk')$: The challenger returns $c' = (c'_0, c'_1, c'_2, c'_3)$, where

$$x, z_1, z_2 \in_R \mathbb{Z}_p,\ \ c'_0 = g_1^x,\ \ c'_1 = (pk')^{(z_1+z_2)},$$

$$h \in_R \mathbb{G}_T,\ \ c'_2 = m \oplus \mathsf{H}_2(h),\ \ c'_3 = h \cdot \hat{e}(g_1, g_2)^{x \cdot (z_1+z_2)}.$$

3. Challenge: The challenger performs in the same way as in Game_1.
4. The challenger performs in the same way as in Game_1 except for the following.
   - $\mathsf{Pextract}(pk, pk', t^*)$: The proxy key is $rk_{pk \xrightarrow{t^*} pk'}$, where

$$s \in_R \mathbb{Z}_p,\ \ q \in_R \mathbb{G}_2,\ \ rk_{pk \xrightarrow{t^*} pk'} = (q, g_2^s \cdot \mathsf{H}_1(0||t)^{-sk}).$$

   - $\mathsf{Preenc}(m, t^*, pk, pk')$: The challenger returns $c' = (c'_0, c'_1, c'_2, c'_3)$, where

$$x, z_1, z_2 \in_R \mathbb{Z}_p,\ \ c'_0 = g_1^x,\ \ c'_1 = (pk')^{(z_1+z_2)},$$

$$h \in_R \mathbb{G}_T,\ \ c'_2 = m \oplus \mathsf{H}_2(h),\ \ c'_3 = h \cdot \hat{e}(g_1, g_2)^{x \cdot (z_1+z_2)}.$$

In this game, the difference with Game_1 is that $(g_2^s, pk', (pk')^s)$ is replaced with $(g_2^s, pk', \frac{q}{\mathsf{H}_1(0||t^*)})$ in answering the $\mathsf{Pextract}$ query with the input (pk, pk', t^*), where the adversary is not allowed to issue a KeyGen_2 query with the input pk'. Let θ_2 be the probability that the challenger successfully ends and $b' = b$ in Game_2, then $|\theta_2 - \theta_1|$ is negligible based on the XDH assumption.

Game_3: In this game, the challenger performs in the same way as in Game_2, except for the Challenge Phase: After receiving (m_0, m_1, t^*), the challenger computes the challenge $c = (c_1, c_2, c_3)$ as follows:

$$b \in_R \{0, 1\},\ \ r \in_R \mathbb{Z}_p,\ \ c_1 = g_1^r,\ \ h, h_1 \in_R \mathbb{G}_T,\ \ c_2 = m_b \oplus \mathsf{H}_2(h),\ \ c_3 = h_1.$$

Compared with Game_2, the challenger faithfully answers the oracle queries except that the following event E_1 occurs: there is a oracle query to H_2 with the input $\frac{h_1}{\hat{e}(g_1, \mathsf{H}_1(0||t))^{u \cdot r}}$. We first describe the following claim.

Claim. The probability $\Pr[E_1]$ is negligible based on the Co-BDH assumption.

The proof of this claim is straightforward by constructing a Co-BDH solver as follows: Given a Co-BDH challenge $(g_1, g_2, g_1^a, g_1^b, g_2^c)$, set $pk = g_1^a$, $g_1^r = g_1^b$, and $\mathsf{H}_1(0||t^*) = g_2^c$. We omit the details here. Let θ_3 be the probability that the challenger successfully ends and $b' = b$ in Game_3. Since $h \in_R \mathbb{G}_T$ and H_2 is a random oracle, we can conclude that $\theta_3 = \frac{1}{2(q_1+1)}$. From Claim 1, we have $|\theta_2 - \theta_3|$ is negligible.

We have shown that the following values are negligible: $|\theta_1 - \theta_2|$, $|\theta_2 - \theta_3|$. Hence, $|\theta_1 - \theta_3| = |\theta_1 - \frac{1}{2(q_1+1)}|$ is negligible. Recall that $|\delta_0 - \frac{1}{2}| = \epsilon$, $\delta_1 = \delta_0$, $\theta_1 = \frac{\delta_1}{q_1+1}$, we have $\frac{\epsilon}{q_1+1}$ is negligible so that ϵ is also negligible. The lemma now follows. □

Lemma 2. *The proposed type-based proxy re-encryption scheme achieves ciphertext privacy unconditionally.*

Proof sketch. Given a message m, the ciphertext generated by $\mathsf{Encrypt}_2$ is $c = (c_0, c_1, c_2, c_3)$, where

$$x, y \in_R \mathbb{Z}_p,\ c_0 = g_1^x,\ c_1 = g_2^{v \cdot y},\ w \in_R \mathbb{G}_T,\ c_2 = m \oplus \mathsf{H}_2(w),\ c_3 = w \cdot \hat{e}(g_1, g_2)^{x \cdot y},$$

while the re-encrypted ciphertext is $c' = (c'_0, c'_1, c'_2, c'_3)$, where

$$r, z \in_R \mathbb{Z}_p, c'_0 = g_1^r, c'_1 = g_2^{v \cdot (s+z)}, h \in_R \mathbb{G}_T, c'_2 = m \oplus \mathsf{H}_2(h), c'_3 = h \cdot \hat{e}(g_1, g_2)^{r \cdot (s+z)}.$$

Since s is a fixed value, the statistical distance between the distributions of c and c' is 0. The lemma now follows. □

5 CCA-secure scheme without ciphertext privacy

In this section we propose a new type-based proxy re-encryption scheme which is IND-PR-CCA secure based on the BDH and the KE assumptions in the random oracle model. However, the scheme does not achieve ciphertext privacy.

5.1 Description of the Scheme

The delegator's type-based PKE scheme. The delegator uses the following type-based PKE scheme $(\mathsf{Setup}_1, \mathsf{KeyGen}_1, \mathsf{Encrypt}_1, \mathsf{Decrypt}_1)$.

1. $\mathsf{Setup}_1(k)$: This algorithm generates two cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p, a generator g of $\mathbb{G}$, a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, and three hash functions

$$\mathsf{H}_1 : \{0,1\}^* \to \mathbb{G},\ \mathsf{H}_2 : \{0,1\}^* \to \mathbb{Z}_p,\ \mathsf{H}_3 : \{0,1\}^* \to \{0,1\}^\ell,$$

where ℓ is a polynomial of k and $\{0,1\}^\ell$ is the plaintext space. The public parameter is denoted as $params_1 = (\mathbb{G}, \mathbb{G}_T, p, g, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3, \hat{e}, \ell)$, and we further assume the type information is $t \in \{0,1\}^*$.

2. $\mathsf{KeyGen}_1(k)$: This algorithm outputs a key pair (pk, sk) where $u \in_R \mathbb{Z}_p$, $pk = g^u$, and $sk = u$.
3. $\mathsf{Encrypt}_1(m, t, pk)$: This algorithm outputs the ciphertext $c = (c_1, c_2, c_3, c_4)$, where

$$h \in \mathbb{G}_T,\ \ c_1 = g^{\mathsf{H}_2(m||h)},\ \ c_2 = h \cdot \hat{e}(pk, \mathsf{H}_1(0||t))^{\mathsf{H}_2(m||h)},$$

$$c_3 = m \oplus \mathsf{H}_3(h),\ \ c_4 = \mathsf{H}_1(1||c_1||c_2||c_3)^{\mathsf{H}_2(m||h)}.$$

4. $\mathsf{Decrypt}_1(c, t, sk)$: This algorithm recovers the plaintext as follows:
   (a) Verify $\hat{e}(c_1, \mathsf{H}_1(1||c_1||c_2||c_3)) = \hat{e}(g, c_4)$.
   (b) Compute $h = \frac{c_2}{\hat{e}(\mathsf{H}_1(0||t), c_1)^{sk}}$ and $m = c_3 \oplus \mathsf{H}_3(h)$.
   (c) Verify $c_1 = g^{\mathsf{H}_2(m||h)}$.
   (d) Return m.
   During decryption, if any of the verifications fails, the algorithm returns an error symbol $\perp$.

The delegatee's PKE scheme. Suppose $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ is a PKE scheme which has the plaintext space $\mathbb{Z}_p$. The delegatees use a PKE scheme $(\mathsf{Setup}_2, \mathsf{KeyGen}_2, \mathsf{Encrypt}_2, \mathsf{Decrypt}_2)$.

1. $\mathsf{Setup}_2(k)$: Output $params_2 = (params, \mathbb{G}, \mathbb{G}_T, p, g, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3, \hat{e}, \ell)$ where $params$ is the public parameter generated by $\mathsf{Setup}(k)$.
2. $\mathsf{KeyGen}_2(k)$: Output a key pair (pk, sk) which is the output of $\mathsf{KeyGen}(k)$.
3. $\mathsf{Encrypt}_2(m, pk)$: This algorithm outputs the ciphertext $c = (c_{-1}, c_0, c_1, c_2, c_3)$, where

$$w \in \mathbb{G}_T, x, y \in_R \mathbb{Z}_p, c_{-1} = \mathsf{Encrypt}(x, pk), c_0 = \mathsf{Encrypt}(y, pk), c_1 = g^{\mathsf{H}_2(m||w)},$$

$$c_2 = w \cdot \hat{e}(g^{\mathsf{H}_2(m||w)}, \mathsf{H}_1(2||x||c_{-1}||pk) \cdot \mathsf{H}_1(2||y||c_0||pk)),\ \ c_3 = m \oplus \mathsf{H}_3(w).$$

4. $\mathsf{Decrypt}_2(c, sk)$: This algorithm recovers the plaintext as follows:
   (a) Compute $w = \frac{c_2}{\hat{e}(c_1, \mathsf{H}_1(2||\mathsf{Decrypt}(c_{-1}, sk)||c_{-1}||pk) \cdot \mathsf{H}_1(2||\mathsf{Decrypt}(c_0, sk)||c_0||pk))}$ and $m = c_3 \oplus \mathsf{H}_3(w)$.
   (b) Verify $c_1 = g^{\mathsf{H}_2(m||w)}$.
   (c) Return m.
   During decryption, if any of the verifications fails, the algorithm returns an error symbol $\perp$.

The delegation algorithms. Suppose the delegator has a key pair (pk, sk). Suppose a delegatee has the key pair (pk', sk'). The algorithms $\mathsf{Pextract}$ and Preenc are defined as follows.

– $\mathsf{Pextract}(pk, pk', t, sk)$: The proxy key is $rk_{pk \xrightarrow{t} pk'}$, where

$$s_1 \in_R \mathbb{Z}_p,\ \ s_2 = \mathsf{Encrypt}(s_1, pk'),$$

$$rk_{pk \xrightarrow{t} pk'} = (s_2, \mathsf{H}_1(2||s_1||s_2||pk') \cdot \mathsf{H}_1(0||t)^{-sk}).$$

– $\mathsf{Preenc}(c, t, rk_{pk \xrightarrow{t} pk'})$: Given a ciphertext $c = (c_1, c_2, c_3, c_4)$, where

$$c_1 = g^{\mathsf{H}_2(m||h)},\ \ c_2 = h \cdot \hat{e}(pk, \mathsf{H}_1(0||t))^{\mathsf{H}_2(m||h)},$$

$$c_3 = m \oplus \mathsf{H}_3(h),\ \ c_4 = \mathsf{H}_1(1||c_1||c_2||c_3)^{\mathsf{H}_2(m||h)}.$$

this algorithm first verifies $\hat{e}(c_1, \mathsf{H}_1(1||c_1||c_2||c_3)) = \hat{e}(g, c_4)$. If the verification passes, it computes a new ciphertext $c' = (c'_{-1}, c'_0, c'_1, c'_2, c'_3)$, where

$$r \in_R \mathbb{Z}_p,\ \ c'_{-1} = \mathsf{Encrypt}(r, pk'),\ \ c'_0 = s_2,\ \ c'_1 = c_1,\ \ c'_3 = c_3,$$

$$\begin{aligned} c'_2 &= c_2 \cdot \hat{e}(c'_1, \mathsf{H}_1(2||r||c'_{-1}||pk') \cdot \mathsf{H}_1(2||s_1||s_2||pk') \cdot \mathsf{H}_1(0||t)^{-sk}) \\ &= h \cdot \hat{e}(pk, \mathsf{H}_1(0||t))^{\mathsf{H}_2(m||h)} \cdot \hat{e}(g^{\mathsf{H}_2(m||h)}, \mathsf{H}_1(2||r||c'_{-1}||pk') \\ &\quad \cdot \mathsf{H}_1(2||s_1||s_2||pk') \cdot \mathsf{H}_1(0||t)^{-sk}) \\ &= h \cdot \hat{e}(g^{\mathsf{H}_2(m||h)}, \mathsf{H}_1(2||r||c'_{-1}||pk') \cdot \mathsf{H}_1(2||s_1||s_2||pk')). \end{aligned}$$

Otherwise, it return an error symbol $\perp$.

It is clear that the re-encrypted ciphertext is also a valid ciphertext for the delegatee so that the delegatee can obtain the plaintext m by running $\mathsf{Decrypt}_2$.

5.2 Security Analysis

The proposed scheme is proven to be IND-PR-CCA secure from Lemma 3.

Lemma 3. *The proposed type-based proxy re-encryption scheme is IND-PR-CCA secure based on the BDH assumption and the KE assumptions in the random oracle model, given that* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is deterministic and one-way.*

Proof sketch. We suppose that the total number of queries of the form $0||t$ issued to H_1 in Phase 1 of the IND-PR-CCA game is bounded by integer q_1. Suppose an adversary $\mathcal{A}$ has the advantage ϵ in the IND-PR-CCA game. The security proof is done through a sequence of games [18].

Game_0: In this game, the challenger faithfully answers the oracle queries from $\mathcal{A}$. The challenger simulates the random oracle H_1 as follows: the challenger maintains a list of vectors, each of them containing a request message, an element of $\mathbb{G}$ (the hash-code for this message), and an element of $\mathbb{Z}_p$. After receiving a request message, the challenger first checks its list to see whether the request message is already in the list. If the check succeeds, the challenger returns the stored element of $\mathbb{G}$; otherwise, the challenger returns g^x, where x a randomly chosen element of $\mathbb{Z}_p$, and stores the new vector in the list. The challenger simulates the random oracle H_2 as follows: the challenger maintains a list of vectors, each of them containing a request message and an element of $\mathbb{Z}_p$ (the hash-code for this message). After receiving a request message, the challenger first checks its list to see whether the request message is already in the list. If the

check succeeds, the challenger returns the stored element of $\mathbb{Z}_p$; otherwise, the challenger returns y which is a randomly chosen element of $\mathbb{Z}_p$, and stores the new vector in the list. The challenger simulates the random oracle $\mathsf{H_3}$ as follows: the challenger maintains a list of vectors, each of them containing a request message and an element of $\{0,1\}^\ell$ (the hash-code for this message). After receiving a request message, the challenger first checks its list to see whether the request message is already in the list. If the check succeeds, the challenger returns the stored element of $\{0,1\}^\ell$; otherwise, the challenger returns z which is a randomly chosen element of $\{0,1\}^\ell$, and stores the new vector in the list. Let $\delta_0 = \Pr[b' = b]$ in $\mathsf{Game_0}$, as we assumed at the beginning, $|\delta_0 - \frac{1}{2}| = \epsilon$.

$\mathsf{Game_1}$: In this game, the challenger performs in the same way as in $\mathsf{Game_0}$, except the following phases.

- Phase 1: The challenger randomly selects $\tau \in \{1, 2, \cdots, q_1 + 1\}$. If $\tau \leq q_1$, suppose the τ-th input to $\mathsf{H_1}$ is $0||t'$. The challenger faithfully answers the oracle queries.
- Challenge: After receiving (m_0, m_1, t^*) from the adversary, the challenger aborts if one of the following events occurs:
  - $0||t^*$ has been queried to $\mathsf{H_1}$ but is not the τ-th query.
  - $0||t^*$ has not been queried to $\mathsf{H_1}$ but $\tau \leq q_1$.

Note that if the challenger does not abort and the τ-th queried has been made in Phase 1, then $t' = t^*$. The probability that the challenger successfully ends is (at least) $\frac{1}{q_1+1}$. Let $\delta_1 = \Pr[b' = b]$ given that the challenger successfully ends, in which case $\delta_1 = \delta_0$. Let θ_1 be the probability that the challenger successfully ends and $b' = b$ in $\mathsf{Game_1}$, then we have $\theta_1 = \frac{\delta_1}{q_1+1}$.

$\mathsf{Game_2}$: In this game, the challenger performs in the same way as in $\mathsf{Game_1}$ except that that collision occurs to $\mathsf{H_1}$, $\mathsf{H_2}$, or $\mathsf{H_3}$. Since these hash functions are modeled as random oracles, therefore the probability a collision occurs to any of them is negligible. Let θ_2 be the probability that the challenger successfully ends and $b' = b$ in $\mathsf{Game_2}$, then we have $|\theta_1 - \theta_2| = \epsilon_2$ is negligible.

$\mathsf{Game_3}$: In this game, the challenger performs in the same way as in $\mathsf{Game_2}$ except for answering $\mathsf{Decrypt_2}$ as follows: Given the input $(c_{-1}, c_0, c_1, c_2, c_3)$, the challenger returns $\perp$ if the following check fails: There are not $\mathsf{H_1}$ queries with the input $2||x'||c_{-1}||pk'$ and $2||y'||c_0||pk'$ to such that $c_0 = \mathsf{Encrypt_1}(y', pk')$, $w' = \frac{c_2}{\hat{e}(c_1, \mathsf{H_1}(2||x'||c_{-1}||pk') \cdot \mathsf{H_1}(2||y'||c_0||pk'))}$, $m' = c_3 \oplus \mathsf{H_3}(w')$, and $c_1 = g^{\mathsf{H_2}(m'||w')}$. Otherwise, the challenger returns m'. Let θ_3 be the probability that the challenger successfully ends and $b' = b$ in $\mathsf{Game_3}$, then we have $|\theta_2 - \theta_3| = \epsilon_3$ is negligible because $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ is deterministic and the hash functions are random oracles.

$\mathsf{Game_4}$: The challenger performs as in the same way as in $\mathsf{Game_3}$ except for answering $\mathsf{Decrypt_1}$ as follows: Given the input (c_1, c_2, c_3, c_4), the challenger returns $\perp$ if the following check fails: There is a query $m'||h'$ to $\mathsf{H_2}$ such that $c_1 = g^{\mathsf{H_2}(m'||h')}$, $m' = c_3 \oplus \mathsf{H_3}(\frac{c_2}{\hat{e}(\mathsf{H_1}(0||t), pk)^{\mathsf{H_2}(m'||h')}})$, and $c_4 = \mathsf{H_1}(1||c_1||c_2||c_3)^{\mathsf{H_2}(m'||h')}$. Otherwise, the challenger returns m'. Let θ_4 be the probability that the challenger successfully ends and $b' = b$ in $\mathsf{Game_4}$, then we have $|\theta_3 - \theta_4| = \epsilon_4$ is negligible because the hash functions are random oracles.

Game_5: The challenger performs in the same way as in Game_4 except for the following. The challenger keeps a list of vectors of the form $(pk, pk', rk_{pk \xrightarrow{t^*} pk'}, h_1)$ and answers $\mathsf{Pextract}$ and Preenc as follows.

- $\mathsf{Pextract}(pk, pk', t^*)$: If the proxy key exists, the challenger returns it. Otherwise, the challenger returns the proxy key $rk_{pk \xrightarrow{t^*} pk'} = (s_2, h_1)$, where

$$s_1 \in_R \mathbb{Z}_p,\ \ s_2 = \mathsf{Encrypt}(s_1, pk'),\ \ h_1 \in_R \mathbb{G},\ \ h_2 = \mathsf{H}_1(2||s_1||s_2||pk'),$$

  and adds $(pk, pk', rk_{pk \xrightarrow{t^*} pk'}, h_1)$ to the list.
- $\mathsf{Preenc}(c, t^*, pk, pk')$: The challenger checks whether a vector of the form

$$(pk, pk', rk_{pk \xrightarrow{t^*} pk'}, h_1)$$

  exists. If not, the challenger generates $rk_{pk \xrightarrow{t^*} pk'} = (s_2, h_1)$, where

$$s_1 \in_R \mathbb{Z}_p,\ \ s_2 = \mathsf{Encrypt}(s_1, pk'),\ \ h_1 \in_R \mathbb{G},\ \ h_2 = \mathsf{H}_1(2||s_1||s_2||pk'),$$

  and adds $(pk, pk', rk_{pk \xrightarrow{t^*} pk'}, h_1)$ to the list. If $c = c_b$, then the challenger returns $c' = (c'_{-1}, c'_0, c'_1, c'_2, c'_3)$, where

$$r \in_R \mathbb{Z}_p,\ c'_{-1} = \mathsf{Encrypt}(r, pk'),\ c'_0 = s_2,\ c'_1 = c_1,\ c'_3 = c_3,\ c'_2 = c_2 \cdot \hat{e}(c'_1, h_1).$$

  Otherwise, the challenger verifies

$$\hat{e}(c_1, \mathsf{H}_1(1||c_1||c_2||c_3)) = \hat{e}(g, c_4).$$

  If the verification passes, the challenger extracts α such that $c_1 = g^\alpha$, returns the re-encrypted ciphertext as

$$r \in_R \mathbb{Z}_p,\ \ c'_{-1} = \mathsf{Encrypt}(r, pk'),\ \ c'_0 = s_2,\ \ c'_1 = c_1, h_3 = \mathsf{H}_1(2||r||c'_{-1}||pk'),$$
$$c'_2 = c_2 \cdot \hat{e}(c'_1, h_2 \cdot h_3) \cdot \hat{e}(pk, \mathsf{H}_1(0||t^*))^{-\alpha},\ \ c'_3 = c_3.$$

Game_5 differs from Game_4 only when one of the following events occurs:

1. For some pk', the adversary issues a H_1 query with the input $2||s_1|| * ||pk'$ for any string $*$, but pk' has not been issued to KeyGen_2.
2. For some pk', the adversary issues a H_1 query with the input $2||r|| * ||pk'$ for any string $*$, but pk' has not been issued to KeyGen_2.
3. The challenger cannot extract α for $c_1 = g^\alpha$.

From the difference lemma [18], we have $|\delta_5 - \delta_4| \leq \epsilon_5$ which is negligible in the random oracle model, given that $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ is one-way and the KE assumption is true for $\mathbb{G}$.

Game_6: In this game, the challenger performs in the same way as in Game_5, except for the Challenge Phase: After receiving (m_0, m_1, t^*), the challenger computes the challenge $c = (c_1, c_2, c_3, c_4)$ as follows:

$$b \in_R \{0, 1\},\ \ y \in_R \mathbb{Z}_p,\ \ h, h_1 \in_R \mathbb{G}_T,$$

$$c_1 \in_R \mathbb{G},\ c_2 = h_1,\ c_3 = m_b \oplus \mathsf{H}_3(h),\ g^y = \mathsf{H}_1(1||c_1||c_2||c_3),\ c_4 = c_1^y.$$

Compared with Game_5, the challenger faithfully answers the oracle queries except that the following event E_1 occurs: there is a oracle query to H_3 with the input $\frac{h_1}{\hat{e}(pk,\mathsf{H}_1(0||t))^{\mathsf{H}_2(m_b||h)}}$. We first describe the following claim.

Claim. The probability $\Pr[E_1]$ is negligible based on the BDH assumption.

The proof of this claim is straightforward, so that we omit the details here. Let θ_6 be the probability that the challenger successfully ends and $b' = b$ in Game_6. Since $h \in_R \mathbb{G}_T$ and H_3 is a random oracle, we can conclude that $\theta_6 = \frac{1}{2(q_1+1)}$. From Claim 2, we have $|\theta_5 - \theta_6|$ is negligible.

We have shown that the following values are negligible: $|\theta_1 - \theta_2|$, $|\theta_2 - \theta_3|$, $|\theta_3 - \theta_4|$, $|\theta_4 - \theta_5|$, and $|\theta_5 - \theta_6|$. Hence, $|\theta_1 - \theta_6| = |\theta_1 - \frac{1}{2(q_1+1)}|$ is negligible. Recall that $|\delta_0 - \frac{1}{2}| = \epsilon$, $\delta_1 = \delta_0$, $\theta_1 = \frac{\delta_1}{q_1+1}$, we have $\frac{\epsilon}{q_1+1}$ is negligible so that ϵ is also negligible. The lemma now follows. □

6 Conclusion

In this paper we have introduced the concept of type-based proxy re-encryption to address the inefficiency issues of traditional proxy re-encryption schemes in practical applications. We have also proposed two schemes which are IND-PR-CPA secure and IND-PR-CCA secure respectively. The IND-PR-CPA secure scheme also achieves ciphertext privacy (which means that a re-encrypted ciphertext is indistinguishable from a normal ciphertext for the delegatee), but the IND-PR-CCA secure scheme does not achieve this attribute. Designing an IND-PR-CCA scheme with ciphertext privacy is left as an open problem. The security model proposed in this paper is particularly designed for traditional PKE schemes, hence, it is interesting to extend it to the ID-based setting.

References

1. G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, 9(1):1–30, 2006.
2. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO 1998*, volume 1462 of *LNCS*, pages 26–45. Springer, 1998.
3. M. Bellare and A. Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In M. K. Franklin, editor, *Advances in Cryptology — CRYPTO 2004*, volume 3152 of *LNCS*, pages 273–289. Springer, 2004.
4. M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73. ACM Press, 1993.
5. M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In K. Nyberg, editor, *Advances in Cryptology — EUROCRYPT '98*, volume 1403 of *LNCS*, pages 127–144. Springer, 1998.

6. D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. In J. Kilian, editor, *Advances in Cryptology — CRYPTO '01*, volume 2139 of *LNCS*, pages 213–229. Springer, 2001.
7. N. Borisov and S. Mitra. Restricted queries over an encrypted index with applications to regulatory compliance. In S. M. Bellovin, R. Gennaro, A. D. Keromytis, and M. Yung, editors, *Applied Cryptography and Network Security, 6th International Conference, ACNS 2008, New York, NY, USA, June 3-6, 2008. Proceedings*, volume 5037 of *LNCS*, pages 373–391, 2008.
8. J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact E-Cash. In R. Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 302–321. Springer, 2005.
9. Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In J. Feigenbaum, editor, *Advances in Cryptology — CRYPTO 1991*, volume 576 of *LNCS*, pages 445–456. Springer, 1991.
10. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology — CRYPTO '84*, volume 196 of *LNCS*, pages 10–18. Springer, 1985.
11. M. Green and G. Ateniese. Identity-based proxy re-encryption. In J. Katz and M. Yung, editors, *Applied Cryptography and Network Security, 5th International Conference*, volume 4521 of *LNCS*, pages 288–306. Springer, 2007.
12. G. Seroussi I. F. Blake and N. P. Smart. *Elliptic Curves in Cryptography*. Cambridge University Press, 1999.
13. A. Ivan and Y. Dodis. Proxy cryptography revisited. In *Proceedings of the Network and Distributed System Security Symposium*. The Internet Society, 2003.
14. Markus Jakobsson. On quorum controlled asymmetric proxy re-encryption. In H. Imai and Y. Zheng, editors, *Public Key Cryptography, Second International Workshop on Practice and Theory in Public Key Cryptography*, volume 1560 of *LNCS*, pages 112–121. Springer, 1999.
15. M. Mambo and E. Okamoto. Proxy cryptosystems: Delegation of the power to decrypt ciphertexts. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, E80-A(1):54–63, 1997.
16. T. Matsuo. Proxy re-encryption systems for identity-based encryption. In T. Takagi, T. Okamoto, E. Okamoto, and T. Okamoto, editors, *Pairing-Based Cryptography — Pairing 2007*, volume 4575 of *LNCS*, pages 247–267. Springer, 2007.
17. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.
18. V. Shoup. Sequences of games: a tool for taming complexity in security proofs. http://shoup.net/papers/, 2006.
19. L. Wang, Z. Cao, T. Okamoto, Y. Miao, and E. Okamoto. Authorization-Limited Transformation-Free Proxy Cryptosystems and Their Security Analyses*. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, (1):106–114, 2006.
20. L. Zhou, M. A. Marsh, F. B. Schneider, and A. Redz. Distributed blinding for distributed elgamal re-encryption. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, pages 824–824. IEEE Computer Society, 2005.